

1111111111	NNN	NNN	SSSSSSSSSSSS	TTTTTTTTTTTTTT	AAAAAAAAAA	LLL	
1111111111	NNN	NNN	SSSSSSSSSSSS	TTTTTTTTTTTTTT	AAAAAAAAAA	LLL	
1111111111	NNN	NNN	SSSSSSSSSSSS	TTTTTTTTTTTTTT	AAAAAAAAAA	LLL	
111	NNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNNNNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNNNNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNNNNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNN	NNN	NNNNNN	SSSSSSSSSS	AAA	AAA	LLL
111	NNN	NNN	NNNNNN	SSSSSSSSSS	AAA	AAA	LLL
111	NNN	NNN	NNNNNN	SSSSSSSSSS	AAA	AAA	LLL
111	NNN	NNNNNN	SSS	TTT	AAAAAAAAAAAAAAAA	LLL	
111	NNN	NNNNNN	SSS	TTT	AAAAAAAAAAAAAAAA	LLL	
111	NNN	NNNNNN	SSS	TTT	AAAAAAAAAAAAAAAA	LLL	
111	NNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNN	NNN	SSS	TTT	AAA	AAA	LLL
111	NNN	NNN	SSSSSSSSSSSS	TTT	AAA	AAA	LLLLLLLLLLLLLLLL
1111111111	NNN	NNN	SSSSSSSSSSSS	TTT	AAA	AAA	LLLLLLLLLLLLLLLL
1111111111	NNN	NNN	SSSSSSSSSSSS	TTT	AAA	AAA	LLLLLLLLLLLLLLLL

• • • •

[illegible]

```
1 0001 0 MODULE INSLIST ( ! Process /LIST and /FULL qualifiers
2 0002 0 IDENT = 'V04-000',
3 0003 0 ADDRESSING_MODE(EXTERNAL = GENERAL)
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1 FACILITY: Install
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 Print the contents of a KFE entry or of all the entries.
37 0037 1
38 0038 1 ENVIRONMENT:
39 0039 1
40 0040 1 VAX/VMS operating system.
41 0041 1
42 0042 1 AUTHOR: Bob Grosso, April 1983
43 0043 1
44 0044 1 Modified by:
45 0045 1
46 0046 1 V03-013 MSH0061 Michael S. Harvey 5-Jul-1984
47 0047 1 List EXECUTE_ONLY attribute if set for known image.
48 0048 1
49 0049 1 V03-012 MSH0057 Michael S. Harvey 26-Jun-1984
50 0050 1 List WRITEABLE attribute along with all the others.
51 0051 1
52 0052 1 V03-011 MSH0049 Michael S. Harvey 17-May-1984
53 0053 1 Don't output meaningless and inaccurate data for
54 0054 1 non-native mode installed images.
55 0055 1
56 0056 1 V03-010 MSH0037 Michael S. Harvey 26-Apr-1984
57 0057 1 Fall back to hard device name if no volume name
```



```
58      0058 1 1 is available.
59      0059 1
60      0060 1 V03-009 MSH0034 Michael S. Harvey 18-Apr-1984
61      0061 1 Display raw device name string in KFD for /STRUCTURE
62      0062 1 listing so we can see what's really stored there.
63      0063 1
64      0064 1 V03-008 MSH0028 Michael S. Harvey 9-Apr-1984
65      0065 1 List maximum shared count correctly. Also, display
66      0066 1 global section count to help interpret the other
67      0067 1 counts being displayed on /FULL listings.
68      0068 1
69      0069 1 V03-007 MSH0026 Michael S. Harvey 4-Apr-1984
70      0070 1 Recognize when known file database either doesn't
71      0071 1 exist or is empty, and do the right thing when one
72      0072 1 tries to access it.
73      0073 1
74      0074 1 V03-006 MSH0022 Michael S. Harvey 20-Mar-1984
75      0075 1 Convert unconcealed device name, which may be an
76      0076 1 allocation class type device name, into a form that
77      0077 1 does not have the allocation class in it.
78      0078 1
79      0079 1 V03-005 BLS0256 Benn Schreiber 27-Dec-1983
80      0080 1 Clean up buffer handling. Reference all pool from EXEC
81      0081 1 mode, since protected against user mode.
82      0082 1
83      0083 1 V03-004 RPG0004 Bob Grosso 13-Sep-1983
84      0084 1 List WCB info.
85      0085 1 Trim blanks from end of line.
86      0086 1
87      0087 1 V03-003 RPG0003 Bob Grosso July 20, 1983
88      0088 1 Clean up listing format.
89      0089 1 Add /structure listing.
90      0090 1 Print listing from user mode.
91      0091 1
92      0092 1 V03-002 RPG0002 Bob Grosso July 8, 1983
93      0093 1 Bypass printing WCB info.
94      0094 1
95      0095 1 V03-001 RPG0001 Bob Grosso July 7, 1983
96      0096 1 Reduce signalling while in EXEC mode.
97      0097 1
98      0098 1 --
99      0099 1
100     0100 1
101     0101 1 Include files
102     0102 1
103     0103 1
104     0104 1 LIBRARY 'SYS$LIBRARY:LIB.L32'; ! VAX/VMS system definitions
105     0105 1
106     0106 1 REQUIRE 'SRC$:INSPREFIX.REQ';
107     0248 1 REQUIRE 'LIB$:INSDEF.R32';
```

Declarations

```
109 0307 1 XSBTTL 'Declarations';
110 0308 1
111 0309 1 Table of contents
112 0310 1
113 0311 1
114 0312 1 FORWARD ROUTINE
115 0313 1     INS_LIST,
116 0314 1     LIST_KFE_ENTRIES,
117 0315 1     LIST_KFE_ENTRY,
118 0316 1     FORMAT_KFD,
119 0317 1     FORMAT_KFE,
120 0318 1     PRINT_PRIVS,
121 0319 1     FORMAT_LINE,
122 0320 1     TERMINATE_LINE : NOVALUE,
123 0321 1     FORMAT_TERMINATE_LINE : NOVALUE,
124 0322 1     PRINTOUT;
125 0323 1
126 0324 1
127 0325 1 External routines
128 0326 1
129 0327 1
130 0328 1 EXTERNAL ROUTINE
131 0329 1     LIB$GET_VM,
132 0330 1     LIB$FREE_VM,
133 0331 1     LIB$PUT_OUTPUT,
134 0332 1     SYSS$GETDVIW : ADDRESSING_MODE (GENERAL),
135 0333 1     SYSS$FAOL : ADDRESSING_MODE (GENERAL);
136 0334 1
137 0335 1 EXTERNAL ROUTINE
138 0336 1     INS$EXECUTE_IN_EXEC_WITH_R_LOCK;
139 0337 1
140 0338 1 EXTERNAL
141 0339 1     CTL$GL_KNOWNFIL,
142 0340 1     EXE$GL_KNOWN_FILES,
143 0341 1     INSS$GL_CTLMSR : BLOCK [1],
144 0342 1     INSS$OUTTAB : BBLOCK,
145 0343 1     PRV$AB_NAMES;
146 0344 1
147 0345 1 EXTERNAL LITERAL
148 0346 1     INSS$EMPTYLIST,
149 0347 1     INSS$FAILGETVM,
150 0348 1     INSS$NOLIST,
151 0349 1     INSS$NOVER;
152 0350 1
153 0351 1 GLOBAL
154 0352 1     INSS$FAOOUTBUF,
155 0353 1     INSS$FAOBUFDESC : BBLOCK [DSC$C_S_BLN];
156 0354 1
157 0355 1 GLOBAL LITERAL
158 0356 1     INSS$C_FAOBUFLN = 255;
159 0357 1
160 0358 1
161 0359 1 Set up user buffer for copying lists to while in kernel mode
162 0360 1
163 0361 1 OWN
164 0362 1     TMPBUF_LEN,
165 0363 1     TMPBUF;
```

! Traverse structure to list all KFEs
! List one KFE
! Format and print KFD block
! Format and print KFE entry
! Print the ASCII keywords for the bits set in a quadword pr
! Format ASCII output into line buffer.
! Copy line buffer to temporary buffer
! Format then terminate line
! Print the contents of the temporary buffer

! get virtual memory
! return virtual memory

! Get Device Information
! Format ASCII output

! Process pointer to the Known file list pointer block
! Exec pointer to the Known file list pointer block
! INSTALL control flags
! Record output block for output buffer
! ASCII list of privileges

! The Known File List is empty
! Failed to get virtual memory
! There is no Known File List
! Error obtaining file version

! Output buffer
! Descriptor of output buffer

! size of output buffer

! Size of allocated buffer
! Address of allocated buffer

Declarations

```

: 166      0364 1      TMPBUF_PTR : REF $BBLOCK;                ! Point to free buffer space
: 167      0365 1
: 168      0366 1      BIND
: 169      0367 1
: 170      0368 1      Control strings for FAO
: 171      0369 1
: 172      0370 1      FAOCTL_DDT          = $DESCRIPTOR ('!AS!AS'),
: 173      0371 1      FAOCTL_VERSION      = $DESCRIPTOR (';!UW'),
: 174      0372 1      FAOCTL_KFDADR       = $DESCRIPTOR (' List head adr/siz/ref = !XL;!UW;!UW'),
: 175      0373 1      FAOCTL_FILNAM       = $DESCRIPTOR ('!AC'),
: 176      0374 1      FAOCTL_FLAGS        = $DESCRIPTOR ('!AC'),
: 177      0375 1      FAOCTL_KFEADR       = $DESCRIPTOR (' Entry address/size/index = !XL;!UW;!XB'),
: 178      0376 1      FAOCTL_WINDOW       = $DESCRIPTOR (' Window address/size = !XL;!UW'),
: 179      0377 1      FAOCTL_HEADER       = $DESCRIPTOR (' Header address/size = !XL;!UW'),
: 180      0378 1      FAOCTL_USECNT       = $DESCRIPTOR (' Entry access count = !UL'),
: 181      0379 1      FAOCTL_SHRUSECNT    = $DESCRIPTOR (' Current / Maximum shared = !UW / !UW'),
: 182      0380 1      FAOCTL_CMODCURR     = $DESCRIPTOR (' Current shared count = !UW'),
: 183      0381 1      FAOCTL_GBLCNT       = $DESCRIPTOR (' Global section count = !UW'),
: 184      0382 1      FAOCTL_COMPAT_TYP   = $DESCRIPTOR (' Compatability type = !XW'),
: 185      0383 1      FAOCTL_PRIVHD      = $DESCRIPTOR (' Privileges = '),
: 186      0384 1      FAOCTL_PRIVHD2     = $DESCRIPTOR (' '),
: 187      0385 1      FAOCTL_PRIV        = $DESCRIPTOR ('!AC ');
: 188      0386 1

```

```
190 0387 1 XSBTTL 'GET_NUMENTRIES':  
191 0388 1 ROUTINE GET_NUMENTRIES (RETCOUNT) =  
192 0389 2 BEGIN  
193 0390 2 +++  
194 0391 2 FUNCTIONAL DESCRIPTION:  
195 0392 2 .....  
196 0393 2 Return the number of entries to allocate for the listing.  
197 0394 2 .....  
198 0395 2 --  
199 0396 2 MAP  
200 0397 2 RETCOUNT : REF VECTOR[,LONG];  
201 0398 2  
202 0399 2 BIND  
203 0400 2 KFPB = EXESGL_KNOWN_FILES : REF $BBLOCK;  
204 0401 2  
205 0402 2 IF .KFPB EQL 0  
206 0403 2 THEN RETURN INSS_NOLIST;  
207 0404 2  
208 0405 2 IF .KFPB[KFPB$KFDLST] EQL 0  
209 0406 2 THEN RETURN INSS_EMPTYLIST;  
210 0407 2  
211 0408 2 RETCOUNT[0] = .KFPB[KFPB$W_KFDLSTCNT];  
212 0409 2 RETURN TRUE  
213 0410 1 END;
```

```
.TITLE INSLIST  
.IDENT \V04-000\  
.PSECT $SPLITS,NOWRT,NOEXE,2  
53 41 21 53 41 21 00000 P.AAB: .ASCII \!AS!AS\  
00006  
00000006 00008 P.AAA: .BLKB 2  
00000000 0000C .LONG 6  
57 55 21 38 00010 P.AAD: .ADDRESS P.AAB  
00000004 00014 P.AAC: .ASCII \;!UW\  
00000000 00018 P.AAC: .LONG 4  
00000000 00018 P.AAD: .ADDRESS P.AAD  
2F 72 64 61 20 64 61 65 68 20 74 73 69 4C 20 0001C P.AAF: .ASCII \ List head adr/siz/ref = !XL/!UW/!UW\  
21 2F 4C 58 21 20 3D 20 66 65 72 2F 7A 69 73 0002B  
57 55 21 2F 57 55 0003A  
00000024 00040 P.AAE: .LONG 36  
00000000 00044 P.AAF: .ADDRESS P.AAF  
43 41 21 20 20 20 00048 P.AAH: .ASCII \ !AC\  
0004E  
00000006 00050 P.AAG: .BLKB 2  
00000000 00054 P.AAG: .LONG 6  
43 41 21 00058 P.AAJ: .ADDRESS P.AAH  
0005B P.AAJ: .ASCII \!AC\  
0005B P.AAI: .BLKB 1  
00000003 0005C P.AAI: .LONG 3  
00000000 00060 P.AAJ: .ADDRESS P.AAJ  
61 20 79 72 74 6E 45 20 20 20 20 20 20 20 20 00064 P.AAL: .ASCII \ Entry address/size/index = !XL\  
64 6E 69 2F 65 7A 69 73 2F 73 73 65 72 64 64 00073  
4C 58 21 20 3D 20 20 20 20 78 65 00082  
42 58 21 2F 57 55 21 2F 0008C  
00000030 00094 P.AAK: .ASCII \!/UW/!XB\  
00000000 00098 P.AAK: .LONG 48  
00000000 .ADDRESS P.AAL
```


INSLIST
V04-000

GET_NUMENTRIES

F 8
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 Bliss-32 V4.0-742
[INSTAL.SRC]INSLIST.B32;1

Page 6
(3)

20 77 6F 64 6E 69 57 20 20 20 20 20 20 20 20 0009C P.AAN: .ASCII \ Window address/size = !XL\	20 20 20 65 7A 69 73 2F 73 73 65 72 64 64 61 000AB
	000BA
	000C4
	000C8 P.AAM: .ASCII \!UW\
	000CC .LONG 44
	000D0 P.AAP: .ADDRESS P.AAN
	000DF .ASCII \ Header address/size = !XL\
	000EE
	000F8
	000FC P.AAO: .ASCII \!UW\
	00100 .LONG 44
	00104 P.AAR: .ADDRESS P.AAP
	00113 .ASCII \ Entry access count = !UL\
	00122
	0012C P.AAQ: .LONG 40
	00130 .ADDRESS P.AAR
	00134 P.AAT: .ASCII \ Current / Maximum shared = !UW\
	00143
	00152
	0015C .ASCII \ / !UW\
	00162 .BLKB 2
	00164 P.AAS: .LONG 46
	00168 .ADDRESS P.AAT
	0016C P.AAV: .ASCII \ Current shared count\<9>\ = \
	0017B
	0018A
	0018E .ASCII \!UW\
	00191 .BLKB 3
	00194 P.AAU: .LONG 37
	00198 .ADDRESS P.AAV
	0019C P.AAX: .ASCII \ Global section count = !UW\
	001AB
	001BA
	001C4 P.AAW: .LONG 40
	001C8 .ADDRESS P.AAX
	001CC P.AAZ: .ASCII \ Compatability type = !XW\
	001DB
	001EA
	001F4 P.AAY: .LONG 40
	001F8 .ADDRESS P.AAZ
	001FC P.ABB: .ASCII \ Privileges = \
	0020B
	00211 .BLKB 3
	00214 P.ABA: .LONG 21
	00218 .ADDRESS P.ABB
	0021C P.ABD: .ASCII \
	0022B
	00231 .BLKB 3
	00234 P.ABC: .LONG 21
	00238 .ADDRESS P.ABD
	0023C P.ABF: .ASCII \!AC \
	00240 P.ABE: .LONG 4
	00244 .ADDRESS P.ABF
	.PSECT \$OWNS,NOEXE,2


```
00000 TMPBUF_LEN:
      .BLKB 4
00004 TMPBUF: .BLKB 4
00008 TMPBUF_PTR:
      .BLKB 4
      .PSECT $GLOBALS,NOEXE,2
```

```
00000 INSS$FAOOUTBUF::
      .BLKB 4
00004 INSS$FAOBUFDESC::
      .BLKB 8
```

```
INSS$FAOBUFLLEN== 255
FAOCTL_DDT= P.AAA
FAOCTL_VERSION= P.AAC
FAOCTL_KFDADR= P.AAE
FAOCTL_FILNAM= P.AAG
FAOCTL_FLAGS= P.AAI
FAOCTL_KFEADR= P.AAK
FAOCTL_WINDOW= P.AAM
FAOCTL_HEADER= P.AAO
FAOCTL_USECNT= P.AAQ
FAOCTL_SHRUSECNT= P.AAS
FAOCTL_CMODCURR= P.AAU
FAOCTL_GBLCNT= P.AAW
FAOCTL_COMPAT_TYP= P.AAY
FAOCTL_PRIVHD= P.ABA
FAOCTL_PRIVHD2= P.ABC
FAOCTL_PRIV= P.ABE
      .EXTRN LIB$GET_VM, LIB$FREE_VM
      .EXTRN LIB$PUT_OUTPUT, SYSS$GETDVIW
      .EXTRN SYSS$FAO, INSS$EXECUTE_IN_EXEC_WITH_R_LOCK
      .EXTRN CTL$GL_KNOWNFIL
      .EXTRN EXE$GL_KNOWN_FILES
      .EXTRN INSS$GL_CTLMSR, INSS$G_OUTTAB
      .EXTRN PRV$AB_NAMES, INSS$EMPTYLIST
      .EXTRN INSS$FAILGETVM, INSS$NOLIST
      .EXTRN INSS$NOVER
      .PSECT $CODE$,NOWRT,2
```

```
0000 00000 GET_NUMENTRIES:
      .WORD Save nothing
50 00000000G 00 D0 00002 MOVL KFPB, R0
      08 12 00009 BNEQ 1$
50 00000000G 8F D0 0000B MOVL #INSS$_NOLIST, R0
      04 00012 RET
      60 D5 00013 1$: TSTL (R0)
      08 12 00015 BNEQ 2$
50 00000000G 8F D0 00017 MOVL #INSS$_EMPTYLIST, R0
      04 0001E RET
04 BC 0C A0 3C 0001F 2$: MOVZWL 12(R0), @RETCOUNT
      50 01 D0 00024 MOVL #1, R0
      04 00027 RET
```

; Routine Size: 40 bytes, Routine Base: \$CODE\$ + 0000

INSLIST
V04-000

GET_NUMENTRIES

H 8
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 Bliss-32 V4.0-742
[INSTAL.SRC]INSLIST.B32;1

Page (3)

```

215 0411 1 %SBTTL 'INSSLIST';
216 0412 1
217 0413 1 GLOBAL ROUTINE INSSLIST ( KFE ) =
218 0414 2 BEGIN
219 0415 2 ***
220 0416 2
221 0417 2 FUNCTIONAL DESCRIPTION:
222 0418 2
223 0419 2 Print the contents of either a specific KFE or all the KFE's.
224 0420 2
225 0421 2 INPUT:
226 0422 2
227 0423 2 kfe = 0 : List all the KFE entries in all the lists.
228 0424 2 = n : List the KFE entry at address 'n'.
229 0425 2
230 0426 2 IMPLICIT OUTPUT:
231 0427 2
232 0428 2 none
233 0429 2
234 0430 2 ROUTINE VALUE:
235 0431 2
236 0432 2 ---
237 0433 2 LITERAL
238 0434 2 MAXLINLEN = 80,
239 0435 2 NUM_FULL_LINES = 3,
240 0436 2 NUM_STRUC_LINES = 3;
241 0437 2
242 0438 2 LOCAL
243 0439 2 NUM_ENTRIES,
244 0440 2 NUM_LINES,
245 0441 2 CME_ARGLST : VECTOR[2, LONG],
246 0442 2 STATUS;
247 0443 2
248 0444 2
249 0445 2 Initialize output buffer and descriptor
250 0446 2
251 0447 2 CH$FILL (XC' ', INSSC FAOBUFLN, .INSSFAOOUTBUF);
252 0448 2 INSSFAOBUFDSC [DSC$Q_LENGTH] = INSSC FAOBUFLN;
253 0449 2 INSSFAOBUFDSC [DSC$A_POINTER] = .INSSFAOOUTBUF;
254 0450 2
255 0451 2 NUM_ENTRIES = 0;
256 0452 2 CME_ARGLST[0] = 1;
257 0453 2 CME_ARGLST[1] = NUM_ENTRIES;
258 0454 2 STATUS = $CMEXEC(ROOTIN=GET_NUMENTRIES, ARGLST=CME_ARGLST);
259 0455 2 IF .STATUS NEQ TRUE
260 0456 2 THEN BEGIN
261 0457 2 SIGNAL(.STATUS);
262 0458 2 RETURN TRUE
263 0459 2 END;
264 0460 2
265 0461 2 IF .KFE NEQ 0
266 0462 2 THEN
267 0463 2 NUM_ENTRIES = 2; ! KFE and KFD
268 0464 2
269 0465 2 NUM_LINES = 2;
270 0466 2 IF .INSSGL_CTLMSK [INSSV_FULL] THEN NUM_LINES = .NUM_LINES + NUM_FULL_LINES;
271 0467 2 IF .INSSGL_CTLMSK [INSSV_STRUCTURE] THEN NUM_LINES = .NUM_LINES + NUM_STRUC_LINES;

```


! routine INSSLIST

0413
0447
0448
0449
0451
0452
0453
0454

0455
0457
0458
0461

0463
0465
0466

0467
0468
0469

INSLIST
V04-000

INSSLIST

K 8
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 Bliss-32 V4.0-742
[INSTAL.SRC]INSLIST.B32;1

Page 11
(4)

00000000G	00	02	FB	0007F	CALLS	#2, LIB\$GET_VM	
	56	50	DO	00086	MOVL	R0, STATUS	
	13	56	EB	00089	BLBS	STATUS, 6\$	0470
		56	DD	0008C	PUSHL	STATUS	0473
		67	DD	0008E	PUSHL	TMPBUF_LEN	
		01	DD	00090	PUSHL	#1	
	00000000G	8F	DD	00092	PUSHL	#INSS FAILGETVM	
	68	04	FB	00098	CALLS	#4, LIB\$SIGNAL	
	50	01	DO	0009B	MOVL	#1, R0	0474
		04	0009E	RET			
67	20	6E	2C	0009F	MOVC5	#0, (SP), #32, TMPBUF_LEN, @TMPBUF	0477
		04	B7	000A4			
	08	A7	DO	000A6	MOVL	TMPBUF, TMPBUF_PTR	0478
		04	AC	DD	PUSHL	KFE	0481
		04	CF	9F	PUSHAB	INS_LIST	
	00000000G	00	02	FB	CALLS	#2, INS\$EXECUTE_IN_EXEC_WITH_R_LOCK	
		56	50	DO	MOVL	R0, STATUS	
	0000V	CF	00	FB	CALLS	#0, PRINTOUT	0482
			A7	9F	PUSHAB	TMPBUF	0484
		04	57	DD	PUSHL	R7	
	00000000G	00	02	FB	CALLS	#2, LIB\$FREE_VM	
		03	50	E9	BLBC	STATUS, 7\$	
		50	56	DO	MOVL	STATUS, R0	0486
			04	000D3	RET		0487

; Routine Size: 212 bytes, Routine Base: \$CODE\$ + 0028

; 292 0488 1

INS_LIST

```
0489 1 %SBTTL 'INS_LIST';
0490 1
0491 1 ROUTINE INS_LIST ( KFE ) =
0492 2 BEGIN
0493 3 +++
0494 3
0495 3 FUNCTIONAL DESCRIPTION:
0496 3
0497 3     Print the contents of either a specific KFE or all the KFE's.
0498 3
0499 3 INPUT:
0500 3
0501 3     kfe      = 0 : List all the KFE entries in all the lists.
0502 3             = n : List the KFE entry at address 'n'.
0503 3
0504 3 IMPLICIT INPUT:
0505 3
0506 3     ins$gl_ctlmsk : INSTALL control flags determine whether to give an
0507 3                   abbreviated or FULL listing.
0508 3     ins$g_outtab  : Record access block for output stream.
0509 3
0510 3 OUTPUT:
0511 3
0512 3     List the known file image list for a single entry
0513 3     or for every entry in all the lists.
0514 3
0515 3 IMPLICIT OUTPUT:
0516 3
0517 3     none
0518 3
0519 3 ROUTINE VALUE:
0520 3
0521 3 ---
0522 3 LOCAL
0523 3 STATUS;
0524 3
0525 3 +++
0526 3
0527 3     Format and print the contents of the buffer
0528 3
0529 3 ---
0530 3
0531 3 IF .KFE EQL 0
0532 3 THEN
0533 3     STATUS = LIST_KFE_ENTRIES ( )
0534 3 ELSE
0535 3     STATUS = LIST_KFE_ENTRY (.KFE);
0536 3
0537 3
0538 3 RETURN .STATUS;
0539 1 END;
```

! routine INS_LIST

0000 00000 INS_LIST:

INSLIST
V04-000

INS_LIST

M 8
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 Bliss-32 V4.0-742
[INSTAL.SRC]INSLIST.832;1

Page 13
(5)

	04	AC	D5	00002	WORD	Save nothing
		06	12	00005	ISTL	KFE
0000V	CF	00	FB	00007	BNEQ	1\$
			04	0000C	CALLS	#0, LIST_KFE_ENTRIES
	04	AC	DD	0000D	RET	
0000V	CF	01	FB	00010	PUSHL	KFE
			04	00015	CALLS	#1, LIST_KFE_ENTRY
					RET	

: 0491
: 0531
: 0533
: 0535
: 0539

; Routine Size: 22 bytes, Routine Base: \$CODE\$ + 00FC

: 345 0540 1

```
347 0541 1 ROUTINE LIST_KFE_ENTRIES =
348 0542 1 +++
349 0543 1 |
350 0544 1 |
351 0545 1 |---
352 0546 2 BEGIN
353 0547 2 LOCAL
354 0548 2     KFD : REF BBLOCK,
355 0549 2     KFE : REF BBLOCK;
356 0550 2
357 0551 2 BIND
358 0552 2     KFPB = EXESGL_KNOWN_FILES : REF BBLOCK;
359 0553 2
360 0554 2 IF .KFPB EQL 0
361 0555 2 THEN
362 0556 2     BEGIN
363 0557 2     RETURN INSS_NOLIST;
364 0558 2     END;
365 0559 2
366 0560 2 IF .KFPB [KFPB$L_KFDLST] EQL 0
367 0561 2 THEN
368 0562 2     BEGIN
369 0563 2     RETURN INSS_EMPTYLIST;
370 0564 2     END;
371 0565 2
372 0566 2 KFD = .KFPB [KFPB$L_KFDLST];
373 0567 2
374 0568 2 |
375 0569 2 |   Traverse the list of KFDs and format each KFD and all its KFEs.
376 0570 2 |   The KFD is the header block which contains the Device, directory and
377 0571 2 |   file type which several Known File Entries (KFE) share in common.
378 0572 2 |
379 0573 2 WHILE .KFD NEQ 0 DO
380 0574 2     BEGIN
381 0575 2     FORMAT_KFD (.KFD);
382 0576 2     KFE = .KFD [KFD$L_KFELIST];
383 0577 2
384 0578 2     |
385 0579 2     |   Format each KFE in the KFD's ordered list of KFEs
386 0580 2     |
387 0581 2     WHILE .KFE NEQ 0 DO
388 0582 2     BEGIN
389 0583 2     FORMAT_KFE (.KFE);
390 0584 2     KFE = .KFE [KFE$L_KFELINK];
391 0585 2     END;
392 0586 2     ! WHILE traversing KFD's ordered KFE list
393 0587 2     KFD = .KFD [KFD$L_LINK];
394 0588 2     END;
395 0589 2     ! Next KFD
396 0590 2     ! WHILE traversing KFD list
397 0591 2 RETURN TRUE;
397 0591 2 END;
```

000C 00000 LIST_KFE_ENTRIES:

INSLIST
V04-000

INS_LIST

B 9
16-Sep-1984 01:54:25 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:35:38 [INSTAL.SRC]INSLIST.B32;1

Page 15
(6)

50	00000000G	00	D0	00002	.WORD	Save R2,R3	:	0541
		08	12	00009	MOVL	KFPB, R0	:	0554
50	00000000G	8F	D0	0000B	BNEQ	1\$:	
			04	00012	MOVL	#INSS_NOLIST, R0	:	0557
		60	D5	00013	RET		:	
		08	12	00015	TSTL	(R0)	:	0560
50	00000000G	8F	D0	00017	BNEQ	2\$:	
			04	0001E	MOVL	#INSS_EMPTYLIST, R0	:	0563
52		60	D0	0001F	RET		:	
		1F	13	00022	MOVL	(R0), KFD	:	0566
		52	DD	00024	BEQL	6\$:	0573
0000V	CF	01	FB	00026	PUSHL	KFD	:	0575
53		A2	D0	0002B	CALLS	#1, FORMAT_KFD	:	
	04	0D	13	0002F	MOVL	4(KFD), KFE	:	0576
		53	DD	00031	BEQL	5\$:	0581
0000V	CF	01	FB	00033	PUSHL	KFE	:	0583
53		A3	D0	00038	CALLS	#1, FORMAT_KFE	:	
	04	F1	11	0003C	MOVL	4(KFE), KFE	:	0584
52		62	D0	0003E	BRB	4\$:	0581
		DF	11	00041	MOVL	(KFD), KFD	:	0587
50		01	D0	00043	BRB	3\$:	0573
		04	00046	6\$:	MOVL	#1, R0	:	0590
					RET		:	0591

: Routine Size: 71 bytes, Routine Base: \$CODE\$ + 0112

: 398 0592 1


```

: 400 0593 1 ROUTINE LIST_KFE_ENTRY (KFE) =
: 401 0594 1 |+++
: 402 0595 1 |
: 403 0596 1 |
: 404 0597 1 |---
: 405 0598 2 BEGIN
: 406 0599 2 MAP
: 407 0600 2 KFE : REF BBLOCK;
: 408 0601 2
: 409 0602 2 FORMAT_KFD (.KFE [KFE$$_KFD]);
: 410 0603 2
: 411 0604 2 FORMAT_KFE (.KFE);
: 412 0605 2
: 413 0606 2 RETURN TRUE;
: 414 0607 1 END;

```

0004 00000 LIST_KFE_ENTRY:

	52	04	AC	D0	00002	WORD	Save R2	
		0C	A2	DD	00006	MOVL	KFE, R2	
0000V	CF		01	FB	00009	PUSHL	12(R2)	
			52	DD	0000E	CALLS	#1, FORMAT_KFD	
0000V	CF		01	FB	00010	PUSHL	R2	
	50		01	D0	00015	CALLS	#1, FORMAT_KFE	
			04	D0	00018	MOVL	#1, R0	
						RET		

```

: 0593
: 0602
:
: 0604
:
: 0606
: 0607

```

; Routine Size: 25 bytes, Routine Base: \$CODE\$ + 0159

; 415 0608 1

```
417 0609 1 ROUTINE FORMAT_KFD (KFD) =
418 0610 1 +++
419 0611 1
420 0612 1
421 0613 1 ---
422 0614 2 BEGIN
423 0615 2 LITERAL
424 0616 2     INS_C_KFDPADLEN = 40;
425 0617 2
426 0618 2 LOCAL
427 0619 2     DEVNAM : BBLOCK [65],
428 0620 2     NEW DEV : BBLOCK [65],
429 0621 2     DEVNAM_DSC : $BBLOCK [DSC$C_S_BLN],
430 0622 2     DDT_DSC : $BBLOCK [DSC$C_S_BLN],
431 0623 2     ITMLST : VECTOR [4, LONG]
432 0624 2     PRESET ( [0] = DVIS_LOGVOLNAM ^ 16 + 64,
433 0625 2         [3] = 0 ),
434 0626 2
435 0627 2 PAD;
436 0628 2
437 0629 2 MAP
438 0630 2     KFD : REF BBLOCK;
439 0631 2
440 0632 2 TERMINATE_LINE ();           ! Blank line
441 0633 2
442 0634 2 IF .INS$GL_CTLMSK [INSSV_STRUCTURE]
443 0635 2 THEN
444 0636 2     ! For a /STRUCTURE listing, simply display the device name in its
445 0637 2     ! raw form as it was stored in the KFD.
446 0638 2
447 0639 2     BEGIN
448 0640 2     DEVNAM_DSC [DSC$W_LENGTH] = .KFD [KFD$B_DEVLEN];
449 0641 2     DEVNAM_DSC [DSC$A_POINTER] = KFD [KFD$T_DDTSTR];
450 0642 2     END
451 0643 2 ELSE                               ! not /STRUCTURE Listing
452 0644 2     ! Build a device name by extracting it from the DDTSTR field of the KFD
453 0645 2     ! and prefixing it with an underscore. The underscore tells $GETDVI that
454 0646 2     ! this is a device name and not to bother trying to translate the string.
455 0647 2
456 0648 2     BEGIN
457 0649 2     DEVNAM_DSC[DSC$A_POINTER] = DEVNAM;           ! Load addr of devnam string
458 0650 2     CH$WCHAR('C' , .DEVNAM_DSC[DSC$A_POINTER]); ! Device, not a logical name
459 0651 2     CH$MOVE(.KFD[KFD$B_DEVLEN], KFD[KFD$T_DDTSTR], .DEVNAM_DSC[DSC$A_POINTER]+1);
460 0652 2     DEVNAM_DSC[DSC$W_LENGTH] = .KFD[KFD$B_DEVLEN]+1; ! Calculate devnam string length
461 0653 2
462 0654 2
463 0655 2     ! Call GETDVI to convert the device name into the volume's logical name
464 0656 2     ! string. This achieves a less confusing/intimidating device name display
465 0657 2     ! for the users.
466 0658 2
467 0659 2
468 0660 2     ITMLST [1] = NEW DEV;           ! Load output buffer address
469 0661 2     ITMLST [2] = DEVNAM_DSC[DSC$W_LENGTH]; ! Shove length into descriptor
470 0662 2     SYSSGETDVIW (0,0,DEVNAM_DSC,ITMLST,0,0,0,0); ! Convert device name format
471 0663 2
472 0664 2
473 0665 2     ! Build the output descriptor for formatting below. If there was a
```

```

474 0666      ! volume logical name defined, then go ahead and use it. If not, then
475 0667      ! we must use the original device name.
476 0668
477 0669      IF .DEVNAM_DSC[DSC$W_LENGTH] EQL 0
478 0670      THEN
479 0671          BEGIN
480 0672              DEVNAM_DSC [DSC$W_LENGTH] = .KFD [KFD$B_DEVLEN];
481 0673              DEVNAM_DSC [DSC$A_POINTER] = KFD [KFD$T_DDTSTR];
482 0674          END
483 0675      ELSE
484 0676          BEGIN
485 0677              CH$WCHAR(':',NEW_DEV+.DEVNAM_DSC[DSC$W_LENGTH]); ! Add and count colon
486 0678              DEVNAM_DSC[DSC$W_LENGTH] = .DEVNAM_DSC[DSC$W_LENGTH] + 1;
487 0679              DEVNAM_DSC[DSC$A_POINTER] = NEW_DEV; ! Finish descriptor
488 0680          END;
489 0681      END;
490 0682
491 0683      ! Now, format the output line.
492 0684
493 0685      DDT_DSC[DSC$W_LENGTH] = .KFD[KFD$B_DDTSTRLEN] - .KFD[KFD$B_DEVLEN];
494 0686      DDT_DSC[DSC$A_POINTER] = KFD[KFD$T_DDTSTR] + .KFD[KFD$B_DEVLEN];
495 0687      FORMAT_LINE (FAOCTL_DDT, DEVNAM_DSC, DDT_DSC); ! Format the KFD output
496 0688
497 0689      IF .INSSGL_CTLMSK [INSSV_STRUCTURE]
498 0690      THEN
499 0691          BEGIN
500 0692              ! Pad the buffer out to INS_C_KFDPADLEN characters
501 0693
502 0694              PAD = INS_C_KFDPADLEN - (INSSC_FAOBUFLN - .INSSFAOBUFDESC [DSC$W_LENGTH]);
503 0695              IF .PAD LEQ 0
504 0696              THEN
505 0697                  BEGIN
506 0698                      TERMINATE_LINE (); ! Print DDT string on first line
507 0699                      PAD = INS_C_KFDPADLEN;
508 0700                  END;
509 0701
510 0702              INSSFAOBUFDESC [DSC$W_LENGTH] = .INSSFAOBUFDESC [DSC$W_LENGTH] - .PAD; ! length is size left in buffer
511 0703              INSSFAOBUFDESC [DSC$A_POINTER] = .INSSFAOBUFDESC [DSC$A_POINTER] + .PAD;
512 0704
513 0705              FORMAT_TERMINATE_LINE (FAOCTL_KFADR, .KFD,
514 0706              .KFD [KFD$W_SIZE], .KFD [KFD$W_REFCNT]); ! Print KFD info
515 0707          END;
516 0708
517 0709      TERMINATE_LINE (); ! Blank line if /STRUCTURE, else prints DDT string
518 0710
519 0711
520 0712      RETURN TRUE;
521 0713  END;

```

.PSECT SPLITS,NOWRT,NOEXE,2

002C0040	00248 P.ABG:	.LONG	2883648
00000000	0024C	.BYTE	0[8]
	00254	.LONG	0


```
.PSECT $CODE$,NOWRT,2

01FC 00000 FORMAT_KFD:
      58 0000V CF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8 0609
      5E FF58 CE 9E 00007 MOVAB TERMINATE_LINE, R8
      6E 0000' 10 28 0000C MOVAB -168(SP),-SP 0625
      68 00 00 0000C MOVAB #16, P.ABG, ITMLST 0631
      56 04 AC D0 00012 CALLS #0, TERMINATE_LINE 0640
      06 00000000G 00 03 E1 00019 BBC #3, INSSGL_CTLMSK+1, 1$ 0633
      57 0E A6 9A 00021 MOVZBL 14(R6), R7 0640
      1C AE 64 AE 9E 00027 1$: BRB 2$
      50 1C AE D0 0002C MOVAB DEVNAM, DEVNAM_DSC+4 0650
      60 5F 8F 90 00030 MOVL DEVNAM_DSC+4, R0 0651
      57 0E A6 9A 00034 MOVZBL 14(R6), R7 0652
      01 18 A0 11 A6 57 28 00038 MOVAB R7, 17(R6), 1(R0) 0653
      AE 57 01 A1 0003E ADDW3 #1, R7, DEVNAM_DSC 0660
      04 AE 20 AE 9E 00043 MOVAB NEW DEV, ITMLST+4 0661
      08 AE 18 AE 9E 00048 MOVAB DEVNAM_DSC, ITMLST+8 0662
      7E 7C 0004D CLRG -(SP)
      7E 7C 0004F CLRG -(SP)
      10 AE 9F 00051 PUSHAB ITMLST
      2C AE 9F 00054 PUSHAB DEVNAM_DSC
      7E 7C 00057 CLRG -(SP)
      00000000G 00 08 FB 00059 CALLS #8, SYSSGETDVIW 0669
      50 18 AE 3C 00060 MOVZWL DEVNAM_DSC, R0
      18 AE 0B 12 00064 BNEQ 3$ 0672
      1C AE 11 A6 9E 0006A 2$: MOVW R7, DEVNAM_DSC 0673
      20 AE40 0D 11 0006F BRB 4$ 0669
      1C AE 18 AE B6 00076 INCW DEVNAM_DSC 0677
      50 20 AE 9E 00079 MOVAB NEW DEV, DEVNAM_DSC+4 0678
      50 10 A6 9A 0007E 4$: MOVZBL 16(R6), R0 0679
      10 AE 11 A746 9E 00087 SUBW3 R7, R0, DDT_DSC 0685
      14 AE 10 AE 9F 0008D MOVAB 17(R7)[R6],-DDT_DSC+4 0686
      1C AE 9F 00090 PUSHAB DDT_DSC 0687
      0000' 0000' CF 9F 00093 PUSHAB DEVNAM_DSC
      03 FB 00097 PUSHAB FAOCTL-DDT
      03 E1 0009C CALLS #3, FORMAT_LINE
      2F 00000000G 00 03 E1 0009C BBC #3, INSSGL_CTLMSK+1, 6$ 0689
      52 0000' CF 3C 000A4 MOVZWL INSSFAOBUFDESC, PAD 0695
      52 FF29 C2 9E 000A9 MOVAB -215(R2), PAD
      68 06 14 000AE BGTR 5$ 0696
      52 00 FB 000B0 CALLS #0, TERMINATE_LINE 0699
      0000' 52 D0 000B3 MOVL #40, PAD 0700
      0000' CF 52 A2 000B6 5$: SUBW2 PAD, INSSFAOBUFDESC 0703
      CF 52 C0 000BB ADDL2 PAD, INSSFAOBUFDESC+4 0704
      7E 0C A6 3C 000C0 MOVZWL 12(R6), -(SP) 0707
      7E 08 A6 3C 000C4 MOVZWL 8(R6), -(SP)
      56 DD 000C8 PUSHL R6 0706
      0000V 0000' CF 9F 000CA PUSHAB FAOCTL_KFDADR
      68 04 FB 000CE CALLS #4, FORMAT_TERMINATE_LINE
      00 FB 000D3 6$: CALLS #0, TERMINATE_LINE 0710
```

INSLIST
V04-000

INS_LIST

6 9
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 Bliss-32 V4.0-742
[INSTAL.SRC]INSLIST.B32;1

Page 20
(8)

: 0712
: 0713

50

01 D0 000D6
04 000D9

MOVL #1, R0
RET

; Routine Size: 218 bytes. Routine Base: SCODES + 0172

: 522 0714 1

```

524 0715 1 ROUTINE FORMAT_KFE (KFE) =
525 0716 1 ----
526 0717 1 ----
527 0718 1 ----
528 0719 1 ----
529 0720 1 BEGIN
530 0721 1 MAP
531 0722 1 KFE : REF BBLOCK;
532 0723 1
533 0724 1
534 0725 1 Constants for setting file information block to get the file version
535 0726 1 number returned via a call to Q10.
536 0727 1
537 0728 1 LITERAL
538 0729 1 FIB_C_FID = 10;
539 0730 1 FIB_C_DID = 12;
540 0731 1 FIB_S_FID = 8;
541 0732 1 INS_C_CTLFLGSTR = 12;
542 0733 1 INS_C_KFEPADLEN = 20;
543 0734 1
544 0735 1 OWN
545 0736 1 FILVER : LONG; ! Address to return file version
546 0737 1 ATRCTLBLK : BBLOCK [12] ! Attribute control block to get version number from
547 0738 1 PRESET ([ATR$W_SIZE] = ATR$S_FILVER,
548 0739 1 [ATR$W_TYPE] = ATR$C_FILVER, ! request file version
549 0740 1 [ATR$L_ADDR] = FILVER
550 0741 1 );
551 0742 1
552 0743 1 FIB : BBLOCK [FIB_C_DID];
553 0744 1 FIB_DESC : BBLOCK [DSC$C_S_BLN]
554 0745 1 PRESET ([DSC$W_LENGTH] = FIB_C_FID,
555 0746 1 [DSC$A_POINTER] = FIB_T,
556 0747 1
557 0748 1
558 0749 1 Control flag array to translate KFE flags to the ASCII
559 0750 1 to be formatted for output.
560 0751 1
561 0752 1 CTLFLG_ARRAY : VECTOR [2*INS_C_CTLFLGSTR] INITIAL (
562 0753 1 KFESM_OPEN, CSTRING ('Open '),
563 0754 1 KFESM_HDRRES, CSTRING ('Hdr '),
564 0755 1 KFESM_SHARED, CSTRING ('Shar '),
565 0756 1 KFESM_PROCPRIV, CSTRING ('Prv '),
566 0757 1 KFESM_PROTECT, CSTRING ('Prot '),
567 0758 1 KFESM_LIM, CSTRING ('Lnkbl '),
568 0759 1 KFESM_COMPATMOD, CSTRING ('Cmode '),
569 0760 1 KFESM_SHMIDENT, CSTRING ('Shm '),
570 0761 1 KFESM_ACCOUNT, CSTRING ('Acnt '),
571 0762 1 KFESM_NOPURGE, CSTRING ('Nopurg '),
572 0763 1 KFESM_WRITEABLE, CSTRING ('Wrt '),
573 0764 1 KFESM_EXEONLY, CSTRING ('Xonly ')
574 0765 1 );
575 0766 1
576 0767 1
577 0768 1 LOCAL
578 0769 1 FID : BBLOCK [FIB_S_FID];
579 0770 1 FLAGS;
580 0771 1 KFD : REF BBLOCK.

```



```
581 0772 2 PAD, ! Number of blanks to pad after filename
582 0773 QIO STATUS,
583 0774 STATUS,
584 0775 WCB_SHRCNT,
585 0776 WCB_SIZ,
586 0777 WCB : REF BBLOCK;
587 0778
588 0779 KFD = .KFE [KFESL_KFD];
589 0780
590 0781
591 0782 Print the File name
592 0783
593 0784 FORMAT_LINE (FAOCTL_FILNAM, KFE [KFESB_FILNAMLEN]);
594 0785
595 0786 CH$FILL (0, FIB_S_FID, FID); ! zero it out
596 0787 WCB = 0;
597 0788
598 0789 IF .KFE [KFESV_OPEN] ! If installed /OPEN, get info from window control block
599 0790 THEN
600 0791 BEGIN
601 0792 LOCAL
602 0793 FCB : REF BBLOCK;
603 0794
604 0795 WCB = .KFE [KFESL_WCB];
605 0796 IF .WCB NEQ 0
606 0797 THEN
607 0798 BEGIN
608 0799 WCB_SIZ = .WCB [WCB$W_SIZE];
609 0800 WCB_SHRCNT = .WCB [WCB$W_REFCNT] - .KFE [KFESW_GBLSECCNT] - 1; ! Amount of file sharing
610 0801 FCB = .WCB [WCB$W_FCB];
611 0802 IF .FCB LSS 0
612 0803 THEN
613 0804 BEGIN
614 0805 CH$MOVE (FIB_S_FID, FCB [FCB$W_FID], FID)
615 0806 END;
616 0807 END;
617 0808 END
618 0809
619 0810 ELSE
620 0811 CH$MOVE (FIB_S_FID, KFE [KFESW_FID], FID);
621 0812
622 0813
623 0814
624 0815 If we obtained the file id field then the file version can be obtained via
625 0816 a call to QIO.
626 0817
627 0818 IF NOT CH$FAIL (CH$FIND_NOT_CH (FIB_S_FID, FID, 0)) ! See if it is all zeros
628 0819 THEN
629 0820 BEGIN
630 0821 LOCAL
631 0822 CHANNEL : WORD,
632 0823 DEVNAM_DESC : BBLOCK [DSC$C_S_BLN],
633 0824 IOSB : BBLOCK [8];
634 0825
635 0826 CH$MOVE (FIB_S_FID, FID, FIB [FIB$W_FID]);
636 0827
637 0828 make descriptor of device name string
```

```
638 0829 3 !
639 0830 3 DEVNAM_DESC = .KFD [KFD$B_DEVLEN];
640 0831 3 DEVNAM_DESC [DSC$A_POINTER] = KFD [KFD$T_DDTSTR];
641 0832 3
642 0833 3
643 0834 3 ! Assign a channel so QIO can be called to get file version
644 0835 3
645 0836 3 STATUS = $ASSIGN ( DEVNAM = DEVNAM_DESC, CHAN = CHANNEL);
646 0837 3 IF NOT .STATUS THEN RETURN .STATUS;
647 0838 3 FILVER = 0;
648 P 0839 3 QIO_STATUS = $QIO (FUNC = IOS_ACCESS, CHAN = .CHANNEL, ! Get the file version
649 0840 3 IOSB = IOSB, P1 = FIB_DESC, P5 = ATRCTLBLK);
650 0841 3
651 0842 3 EXECUTE ($DASSGN (CHAN = .CHANNEL) ); ! Deassign the channel
652 0843 3
653 0844 3 IF NOT .IOSB
654 0845 3 THEN
655 0846 3 BEGIN
656 0847 3 !
657 0848 3 ! Build a descriptor of the file name which is now in
658 0849 3 ! the output buffer and indicate that the file was not found
659 0850 3
660 0851 3 LITERAL
661 0852 3 DECODED_MSGBUF_LEN = 256,
662 0853 3 ERRFILNAM_BUFLen = 31;
663 0854 3 LOCAL
664 0855 3 DECODED_MSGDSC : BBLOCK [DSC$S_BLN],
665 0856 3 DECODED_MSGBUF : BBLOCK [DECODED_MSGBUF_LEN],
666 0857 3 ERRFILNAM_DSC : BBLOCK [DSC$S_BLN],
667 0858 3 ERRFILNAM_BUF : BBLOCK [ERRFILNAM_BUFLen];
668 0859 3
669 0860 3 ERRFILNAM_DSC [DSC$A_POINTER] = ERRFILNAM_BUF;
670 0861 3 ERRFILNAM_DSC = INSSC_FAOBUFLen - .INSSC_FAOBUFLen [DSC$W_LENGTH];
671 0862 3 IF .ERRFILNAM_DSC [DSC$W_LENGTH] GTR ERRFILNAM_BUFLen
672 0863 3 THEN ERRFILNAM_DSC [DSC$W_LENGTH] = ERRFILNAM_BUFLen;
673 0864 3
674 0865 3 CH$MOVE (.ERRFILNAM_DSC [DSC$W_LENGTH], .INSSC_FAOOUTBUF,
675 0866 3 ERRFILNAM_BUF);
676 0867 3
677 0868 3 DECODED_MSGDSC = DECODED_MSGBUF_LEN;
678 0869 3 DECODED_MSGDSC [DSC$A_POINTER] = DECODED_MSGBUF;
679 0870 3 CH$FILL (0, DECODED_MSGBUF_LEN, DECODED_MSGBUF);
680 0871 3
681 P 0872 3 $GETMSG ( MSGID = INSSC_MOVER,
682 P 0873 3 MSGLEN = DECODED_MSGDSC,
683 0874 3 BUFADR = DECODED_MSGDSC);
684 0875 3
685 0876 3 TERMINATE LINE ();
686 0877 3 FORMAT_TERMINATE_LINE ( DECODED_MSGDSC, ERRFILNAM_DSC);
687 0878 3
688 0879 3 DECODED_MSGDSC = DECODED_MSGBUF_LEN;
689 0880 3 DECODED_MSGDSC [DSC$A_POINTER] = DECODED_MSGBUF;
690 0881 3 CH$FILL (0, DECODED_MSGBUF_LEN, DECODED_MSGBUF);
691 0882 3
692 P 0883 3 $GETMSG ( MSGID = .IOSB,
693 P 0884 3 MSGLEN = DECODED_MSGDSC,
694 0885 3 BUFADR = DECODED_MSGDSC);
```

```

695 0886 4
696 0887 4      FORMAT_LINE ( DECODED_MSGDSC);
697 0888 4      END
698 0889 4
699 0890 4      ELSE
700 0891 4      BEGIN
701 0892 4          FORMAT_LINE (FAOCTL_VERSION, .FILVER ); ! Format the version into output buffer
702 0893 4      END;
703 0894 4      END;
704 0895 4
705 0896 4
706 0897 4      Pad the buffer out to INS_C_KFEPADLEN characters
707 0898 4
708 0899 4      PAD = INS_C_KFEPADLEN - (INSSC_FAOBUFLN - .INSSFAOBUFDESC [DSC$W_LENGTH]);
709 0900 4      IF .PAD LEQ 0
710 0901 4      THEN
711 0902 4      BEGIN
712 0903 4          TERMINATE_LINE ();
713 0904 4          PAD = INS_C_KFEPADLEN;
714 0905 4      END;
715 0906 4
716 0907 4      INSSFAOBUFDESC [DSC$W_LENGTH] = .INSSFAOBUFDESC [DSC$W_LENGTH] - .PAD; !length is size left in buffer
717 0908 4      INSSFAOBUFDESC [DSC$A_POINTER] = .INSSFAOBUFDESC [DSC$A_POINTER] + .PAD;
718 0909 4
719 0910 4
720 0911 4      Decode KFE flags
721 0912 4
722 0913 4      BEGIN
723 0914 4      LOCAL
724 0915 4          BUFLN,
725 0916 4          BUFPTR;
726 0917 4
727 0918 4      BUFLN = .INSSFAOBUFDESC [DSC$W_LENGTH];
728 0919 4      BUFPTR = .INSSFAOBUFDESC [DSC$A_POINTER];
729 0920 4
730 0921 4      FLAGS = .KFE [KFESW_FLAGS];
731 0922 4
732 0923 4
733 0924 4      Search the table, if the mask is set in the composite control
734 0925 4      flags longword, then call FAOL with the corresponding descriptor
735 0926 4
736 0927 4      INCR I FROM 0 TO (2 * INS_C_CTLFLGSTR -1) BY 2 DO
737 0928 4      BEGIN
738 0929 4      BIND
739 0930 4          MASK = CTLFLG_ARRAY [.I],
740 0931 4          CSTRNG = CTLFLG_ARRAY [.I] + 4,
741 0932 4          PADLEN = .CSTRNG : BYTE;
742 0933 4
743 0934 4      IF (.MASK AND .FLAGS) NEQ 0
744 0935 4      THEN
745 0936 4      BEGIN
746 0937 4          FORMAT_LINE (FAOCTL_FLAGS, .CSTRNG);
747 0938 4          BUFLN = .INSSFAOBUFDESC [DSC$W_LENGTH];
748 0939 4          BUFPTR = .INSSFAOBUFDESC [DSC$A_POINTER];
749 0940 4      END
750 0941 4      ELSE
751 0942 4      BEGIN

```

```

752      INSS$FAOBUDESC [DSC$W_LENGTH] = .INSS$FAOBUDESC [DSC$W_LENGTH] - .PADLEN;
753      INSS$FAOBUDESC [DSC$A_POINTER] = .INSS$FAOBUDESC [DSC$A_POINTER] + .PADLEN;
754      END;
755      END;
756      INSS$FAOBUDESC [DSC$W_LENGTH] = .BUFLEN;
757      INSS$FAOBUDESC [DSC$A_POINTER] = .BUFPTR;
758      END;
759      ---
760      Print extra info for a /FULL or /STRUCTURE listing
761      IF .INSS$GL_CTLMSK [INSS$V_FULL]
762      THEN
763      BEGIN
764      TERMINATE_LINE ();          ! Print file name or decoded flags
765      IF .INSS$GL_CTLMSK [INSS$V_STRUCTURE]
766      THEN
767      FORMAT_TERMINATE_LINE (FAOCTL_KFEADR, .KFE
768      .KFE [KFES$W_SIZE], .KFE [KFES$B_HSHIDX]);
769      IF .KFE [KFES$V_COMPATMOD] ! Mark as compatibility mode image
770      THEN
771      FORMAT_TERMINATE_LINE (FAOCTL_COMPAT_TYP, .KFE [KFES$W_AMECOD])
772      ELSE
773      FORMAT_TERMINATE_LINE (FAOCTL_USECNT, .KFE [KFES$L_USECNT]);
774      IF .KFE [KFES$V_OPEN] ! If /OPEN
775      THEN
776      IF .KFE [KFES$V_COMPATMOD]
777      THEN
778      FORMAT_TERMINATE_LINE (FAOCTL_CMODCURR, .WCB_SHRCNT)
779      ELSE
780      FORMAT_TERMINATE_LINE (FAOCTL_SHRUSECNT,
781      .WCB_SHRCNT, .KFE [KFES$W_SHRCNT] - 1);
782      IF .KFE [KFES$V_SHARED] ! If /SHARED
783      THEN
784      FORMAT_TERMINATE_LINE (FAOCTL_GBLCNT, .KFE [KFES$W_GBLSECCNT]);
785      IF (.INSS$GL_CTLMSK [INSS$V_STRUCTURE] AND .WCB NEQ 0) ! If installed /OPEN, print info on window
786      THEN
787      FORMAT_TERMINATE_LINE (FAOCTL_WINDOW, .WCB, .WCB_SIZ);
788      IF (.INSS$GL_CTLMSK [INSS$V_STRUCTURE] AND .KFE [KFES$V_HDRRES]) ! If header resident
789      THEN
790      BEGIN
791      BIND
792      KFRH = .KFE [KFES$L_IMGHDR] - KFRH$C_LENGTH : BBLOCK;
793      FORMAT_TERMINATE_LINE (FAOCTL_HEADER,
794      .KFE [KFES$L_IMGHDR], .KFRH [KFES$W_SIZE]);
795      END;
796      IF .KFE [KFES$V_PROCPRIV]
797      THEN
798      END;
799      END;
800      END;
801      END;
802      END;
803      END;
804      END;
805      END;
806      END;
807      END;
808      END;
```



```

: 809      1000 3      PRINT_PRIVS (KFE [KFESQ_PROCPRIV]);
: 810      1001      END;
: 811      1002      ! Full Listing
: 812      1003      ! If /FULL prints blank line, else prints file name
: 813      1004      TERMINATE_LINE ();
: 814      1005      RETURN TRUE;
: 815      1006      END;
: 816      1007

```

```

.PSECT $SPLITS,NOVRT,NOEXE,2

20 6E 65 70 05 00258 P.ABH: .BYTE 5
      4F 00259 .ASCII \Open \
      04 0025E P.ABI: .BYTE 4
20 72 64 48 0025F .ASCII \Hdr \
      05 00263 P.ABJ: .BYTE 5
20 72 61 68 53 00264 .ASCII \Shar \
      04 00269 P.ABK: .BYTE 4
      50 0026A .ASCII \Prv \
20 74 6F 72 05 0026E P.ABL: .BYTE 5
      50 0026F .ASCII \Prot \
      06 00274 P.ABM: .BYTE 6
20 6C 62 6B 6E 4C 00275 .ASCII \Lnkbl \
      06 0027B P.ABN: .BYTE 6
20 65 64 6F 6D 43 0027C .ASCII \Cmode \
      04 00282 P.ABO: .BYTE 4
      53 00283 .ASCII \Shm \
20 74 6E 63 05 00287 P.ABP: .BYTE 5
      41 00288 .ASCII \Acnt \
      07 0028D P.ABQ: .BYTE 7
20 67 72 75 70 6F 4E 0028E .ASCII \Nopurg \
      04 00295 P.ABR: .BYTE 4
      57 00296 .ASCII \Wrt \
      05 0029A P.ABS: .BYTE 5
79 6C 6E 6F 58 0029B .ASCII \Xonly\

```

```

.PSECT $OWNS,NOEXE,2

0007 0002 0000C FILVER: .BLKB 4
00000000' 00010 ATRCTLBLK:
      .WORD 2, 7
      .ADDRESS FILVER
      00018 .BLKB 4
      0001C FIB: .BLKB 12
000A 00028 FIB_DESC:
      .WORD 10
      00# 0002A .BYTE 0[2]
00000000' 0002C .ADDRESS FIB
00000008 00030 CTLFLG_ARRAY:
      .LONG 8
00000000' 00034 .ADDRESS P.ABH
00000010 00038 .LONG 16
00000000' 0003C .ADDRESS P.ABI
00000020 00040 .LONG 32
00000000' 00044 .ADDRESS P.ABJ

```

```
00000004 00048 .LONG 4
00000000' 0004C .ADDRESS P.ABK
00000001 00050 .LONG 1
00000000' 00054 .ADDRESS P.ABL
00000002 00058 .LONG 2
00000000' 0005C .ADDRESS P.ABM
00000080 00060 .LONG 128
00000000' 00064 .ADDRESS P.ABN
00000040 00068 .LONG 64
00000000' 0006C .ADDRESS P.ABO
00000200 00070 .LONG 512
00000000' 00074 .ADDRESS P.ABP
00000100 00078 .LONG 256
00000000' 0007C .ADDRESS P.ABQ
00000400 00080 .LONG 1024
00000000' 00084 .ADDRESS P.ABR
00000800 00088 .LONG 2048
00000000' 0008C .ADDRESS P.ABS
```

```
.EXTRN SYSS$ASSIGN, SYSS$QIOW
.EXTRN SYSS$DASSGN, SYSS$GETMSG
```

```
.PSECT $CODE$,NOWRT,2
```

```
OFFC 00000 FORMAT_KFE:
```

```
08      00      0000V  5E  FEB4  CE  9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11 : 0715
      58      04  AC  D0 00007 MOVAB -332(SP), SP : 0779
      59      0C  A8  D0 0000B MOVL KFE, R8 : 0784
      36      A8  9F 0000F MOVL 12(R8), KFD : 0786
0000' 0000'  CF  9F 00012 PUSHAB FAOCTL FILNAM : 0787
      6E      02  FB 00016 CALLS #2, FORMAT LINE : 0789
      F8      00  2C 0001B MOVC5 #0, (SP), #0, #8, FID : 0795
      56      56  D4 00022 CLRL WCB : 0796
      6A      10  A8  9E 00024 MOVAB 16(R8), R10 : 0800
      56      03  E1 00028 BBC #3, (R10), 1$ : 0801
      5B      18  A8  D0 0002C MOVL 24(R8), WCB : 0802
      57      25  13 00030 BEQL 2$ : 0805
      50      08  A6  3C 00032 MOVZWL 8(WCB), WCB_SIZ : 0811
      57      0E  A6  3C 00036 MOVZWL 14(WCB), R7 : 0818
      50      12  A8  3C 0003A MOVZWL 18(R8), R0 : 0826
      57      50  C2 0003E SUBL2 R0, R7 : 0830
      50      57  D7 00041 DECL WCB_SHRCNT : 0830
      18      A6  D0 00043 MOVL 24(WCB), FCB : 0830
      0E      08  28 00047 BGEQ 2$ : 0830
      A0      06  11 0004F MOVC3 #8, 36(FCB), FID : 0830
      AD      08  28 00051 1$: MOVC3 #8, 24(R8), FID : 0830
      AD      00  3B 00057 2$: SKPC #0, #8, FID : 0830
      08      02  12 0005C 3$: CLRL R1 : 0830
      51      51  D5 00060 4$: TSTL R1 : 0830
      51      03  12 00062 BNEQ 4$ : 0830
      00FD 31 00064 BRW 10$ : 0830
      08  28 00067 MOVC3 #8, FID, FID+4 : 0830
      0E  A9  9A 0006E MOVZBL 14(KFD), DEVNAM_DESC : 0830
```

	F4	AD	11	A9	9E	00073	MOVAB	17(R9), DEVNAM_DESC+4	0831
				7E	7C	00078	CLRQ	-(SP)	0836
			08	AE	9F	0007A	PUSHAB	CHANNEL	
			FO	AD	9F	0007D	PUSHAB	DEVNAM_DESC	
	00000000G	00		04	FB	00080	CALLS	#4, SYSS\$ASSIGN	
		30		50	E9	00087	BLBC	STATUS, 5\$	0837
			0000'	CF	D4	0008A	CLRL	FILVER	0838
				7E	D4	0008E	CLRL	-(SP)	0840
			0000'	CF	9F	00090	PUSHAB	ATRCTLBLK	
				7E	7C	00094	CLRQ	-(SP)	
				7E	D4	00096	CLRL	-(SP)	
			0000'	CF	9F	00098	PUSHAB	FIB_DESC	
				7E	7C	0009C	CLRQ	-(SP)	
			E8	AD	9F	0009E	PUSHAB	IOSB	
				32	DD	000A1	PUSHL	#50	
		7E	28	AE	3C	000A3	MOVZWL	CHANNEL, -(SP)	
				7E	D4	000A7	CLRL	-(SP)	
	00000000G	00		0C	FB	000A9	CALLS	#12, SYSS\$QIOW	
		7E		6E	3C	000B0	MOVZWL	CHANNEL, -(SP)	0842
	00000000G	00		01	FB	000B3	CALLS	#1, SYSS\$DASSGN	
		01		50	E8	000BA	BLBS	STATUS, 6\$	
					04	000BD	RET		
		03	F8	AD	E9	000BE	BLBC	IOSB, 7\$	0844
				0092	31	000C2	BRW	9\$	
		28	04	AE	9E	000C5	MOVAB	ERRFILNAM_BUF, ERRFILNAM_DSC+4	0860
		24	0000'	CF	3C	000CA	MOVZWL	INSS\$FAOBUFDESC, ERRFILNAM_DSC	0861
24	AE	000000FF	24	AE	C3	000D0	SUBL3	ERRFILNAM_DSC, #255, ERRFILNAM_DSC	
			24	AE	B1	000DA	CMW	ERRFILNAM_DSC, #31	0862
				04	1B	000DE	BLEQU	8\$	
		24	AE	1F	B0	000E0	MOVW	#31, ERRFILNAM_DSC	0863
04	AE	0000'	DF	24	AE	28	MOVCS	ERRFILNAM_DSC, @INSS\$FAOOUTBUF, -	0865
								ERRFILNAM_BUF	
		E0	AD	0100	8F	3C	MOVZWL	#256, DECODED_MSGDSC	0868
		E4	AD	2C	AE	9E	MOVAB	DECODED_MSGBUF, DECODED_MSGDSC+4	0869
0100	8F	00	6E		00	2C	MOVCS	#0, (SPT, #0, #256, DECODED_MSGBUF	0870
				2C	AE	000FE			
			7E	0F	7D	00100	MOVQ	#15, -(SP)	0874
				AD	9F	00103	PUSHAB	DECODED_MSGDSC	
				AD	9F	00106	PUSHAB	DECODED_MSGDSC	
				8F	DD	00109	PUSHL	#INSS\$NOVER	
	00000000G	00		05	FB	0010F	CALLS	#5, SYSS\$GETMSG	
		0000V	CF	00	FB	00116	CALLS	#0, TERMINATE_LINE	0876
				24	AE	9F	PUSHAB	ERRFILNAM_DSC	0877
			E0	AD	9F	0011E	PUSHAB	DECODED_MSGDSC	
		0000V	CF	02	FB	00121	CALLS	#2, FORMAT_TERMINATE_LINE	
		E0	AD	0100	8F	3C	MOVZWL	#256, DECODED_MSGDSC	0879
		E4	AD	2C	AE	9E	MOVAB	DECODED_MSGBUF, DECODED_MSGDSC+4	0880
0100	8F	00	6E		00	2C	MOVCS	#0, (SPT, #0, #256, DECODED_MSGBUF	0881
				2C	AE	00138			
			7E	0F	7D	0013A	MOVQ	#15, -(SP)	0885
				AD	9F	0013D	PUSHAB	DECODED_MSGDSC	
				AD	9F	00140	PUSHAB	DECODED_MSGDSC	
				AD	DD	00143	PUSHL	IOSB	
	00000000G	00		05	FB	00146	CALLS	#5, SYSS\$GETMSG	
			E0	AD	9F	0014D	PUSHAB	DECODED_MSGDSC	0887
		0000V	CF	01	FB	00150	CALLS	#1, FORMAT_LINE	
				0D	11	00155	BRB	10\$	0844

		0000'	CF	DD	00157	98:	PUSHL	FILVER	0892	
		0000'	CF	9F	00158		PUSHAB	FAOCTL_VERSION		
	0000V	CF	02	FB	0015F		CALLS	#2, FORMAT_LINE		
	52	0000'	CF	3C	00164	108:	MOVZWL	INSSFAOBUFDESC, PAD	0899	
	52	FF15	C2	9E	00169		MOVAB	-235(R2), PAD		
			08	14	0016E		BGTR	11\$	0900	
	0000V	CF	00	FB	00170		CALLS	#0, TERMINATE_LINE	0903	
	52		14	DD	00175		MOVL	#20, PAD	0904	
	0000'	CF	52	A2	00178	118:	SUBW2	PAD, INSSFAOBUFDESC	0907	
	0000'	CF	52	CO	0017D		ADDL2	PAD, INSSFAOBUFDESC+4	0908	
	55	0000'	CF	3C	00182		MOVZWL	INSSFAOBUFDESC, BUFLN	0918	
	54	0000'	CF	DD	00187		MOVL	INSSFAOBUFDESC+4, BUFPTR	0919	
	59		6A	3C	0018C		MOVZWL	(R10), FLAGS	0921	
			52	D4	0018F		CLRL	I	0934	
	53	0000'	CF	42	DD	00191	128:	MOVL	CTLFLG_ARRAY+4[I], R3	0932
	59	0000'	CF	42	D3	00197		BITL	CTLFLG_ARRAY[I], FLAGS	0934
			1A	13	0019D		BEQL	13\$		
		0000'	CF	42	DD	0019F	PUSHL	CTLFLG_ARRAY+4[I]	0937	
		0000'	CF	9F	001A4		PUSHAB	FAOCTL_FLAGS		
	0000V	CF	02	FB	001A8		CALLS	#2, FORMAT_LINE		
	55	0000'	CF	3C	001AD		MOVZWL	INSSFAOBUFDESC, BUFLN	0938	
	54	0000'	CF	DD	001B2		MOVL	INSSFAOBUFDESC+4, BUFPTR	0939	
			10	11	001B7		BRB	14\$	0934	
	50		63	9A	001B9	138:	MOVZBL	(R3), R0	0943	
	0000'	CF	50	A2	001BC		SUBW2	R0, INSSFAOBUFDESC		
	50		63	9A	001C1		MOVZBL	(R3), R0	0944	
	0000'	CF	50	CO	001C4		ADDL2	R0, INSSFAOBUFDESC+4		
FFC2	52	0000'	CF	17	F1	001C9	148:	ACBL	#23, #2, I, 12\$	0927
	0000'	CF	55	B0	001CF		MOVW	BUFLN, INSSFAOBUFDESC	0948	
	0000'	CF	54	DD	001D4		MOVL	BUFPTR, INSSFAOBUFDESC+4	0949	
03	00000000G	00	02	EO	001D9		RBS	#2, INSSGL_CTLMSK+1, 15\$	0955	
		00B7	31	001E1		BRW	24\$			
	0000V	CF	00	FB	001E4	158:	CALLS	#0, TERMINATE_LINE	0958	
13	00000000G	00	03	E1	001E9		BBC	#3, INSSGL_CTLMSK+1, 16\$	0960	
	7E	08	A8	9A	001F1		MOVZBL	11(R8), -(SP)	0963	
	7E	08	A8	3C	001F5		MOVZWL	8(R8), -(SP)		
			58	DD	001F9		PUSHL	R8	0962	
	0000'	CF	9F	001FB		PUSHAB	FAOCTL_KFEADR			
	0000V	CF	04	FB	001FF		CALLS	#4, FORMAT_TERMINATE_LINE		
			6A	95	00204	168:	TSTB	(R10)	0965	
			0A	18	00206		BGEQ	17\$		
	7E	2A	A8	3C	00208		MOVZWL	42(R8), -(SP)	0967	
		0000'	CF	9F	0020C		PUSHAB	FAOCTL_COMPAT_TYP		
			07	11	00210		BRB	18\$		
	14	0000'	A8	DD	00212	178:	PUSHL	20(R8)	0969	
		0000'	CF	9F	00215		PUSHAB	FAOCTL_USECNT		
	0000V	CF	02	FB	00219	188:	CALLS	#2, FORMAT_TERMINATE_LINE		
22		6A	03	E1	0021E		BBC	#3, (R10), -20\$	0971	
			6A	95	00222		TSTB	(R10)	0975	
			0D	18	00224		BGEQ	19\$		
			57	DD	00226		PUSHL	WCB_SHRCNT		
	0000'	CF	9F	00228		PUSHAB	FAOCTL_CHMODCURR			
	0000V	CF	02	FB	0022C		CALLS	#2, FORMAT_TERMINATE_LINE		
			11	11	00231		BRB	20\$		
	7E	34	A8	3C	00233	198:	MOVZWL	52(R8), -(SP)	0978	
			6E	D7	00237		DECL	(SP)		
			57	DD	00239		PUSHL	WCB_SHRCNT		

			0000'	CF	9F	0023B	PUSHAB	FAOCTL SHRUSECNT	0977
				03	FB	0023F	CALLS	#3, FORMAT_TERMINATE_LINE	
0D	0000V	CF		05	E1	00244	BBC	#5, (R10), -21\$	0980
		6A	12	A8	3C	00248	MOVZWL	18(R8), -(SP)	0982
		7E	0000'	CF	9F	0024C	PUSHAB	FAOCTL GBLCNT	
	0000V	CF		02	FB	00250	CALLS	#2, FORMAT_TERMINATE_LINE	
32	00000000G	00		03	E1	00255	BBC	#3, INSSGL_CTLMSK+1, -23\$	0984
				56	D5	0025D	TSTL	WCB	
				0D	13	0025F	BEQL	22\$	
			0840	8F	BB	00261	PUSHR	#M<R6,R11>	0986
			0000'	CF	9F	00265	PUSHAB	FAOCTL WINDOW	
	0000V	CF		03	FB	00269	CALLS	#3, FORMAT_TERMINATE_LINE	
19	00000000G	00		03	E1	0026E	BBC	#3, INSSGL_CTLMSK+1, -23\$	0988
15		6A		04	E1	00276	BBC	#4, (R10), -23\$	
50	1C	A8		0C	C3	0027A	SUBL3	#12, 28(R8), R0	0992
		7E	08	A0	3C	0027F	MOVZWL	8(R0), -(SP)	0995
			1C	A8	DD	00283	PUSHL	28(R8)	
			0000'	CF	9F	00286	PUSHAB	FAOCTL HEADER	0994
	0000V	CF		03	FB	0028A	CALLS	#3, FORMAT_TERMINATE_LINE	
08		6A		02	E1	0028F	BBC	#2, (R10), -24\$	0998
			20	A8	9F	00293	PUSHAB	32(R8)	1000
	0000V	CF		01	FB	00296	CALLS	#1, PRINT PRIVS	
	0000V	CF		00	FB	0029B	CALLS	#0, TERMINATE_LINE	1004
		50		01	D0	002A0	MOVL	#1, R0	1006
				04	002A3	RET			1007

; Routine Size: 676 bytes, Routine Base: \$CODE\$ + 024C

INS_LIST

```

818 1008 1
819 1009 1 XSBTTL 'PRINT_PRIVS';
820 1010 1
821 1011 1 ROUTINE PRINT_PRIVS (PRIV_ADR) =
822 1012 1 +++
823 1013 1
824 1014 1 FUNCTIONAL DESCRIPTION:
825 1015 1 Print the ASCII symbol for each privilege bit set in the quadword
826 1016 1 privilege mask, priv_adr.
827 1017 1
828 1018 1 INPUT:
829 1019 1 priv_adr = address of quadword privilege mask
830 1020 1
831 1021 1
832 1022 2 BEGIN
833 1023 2 LOCAL
834 1024 2 PLACE_HLDR,
835 1025 2 PRVS TO PRINT,
836 1026 2 SYMBOL_LEN,
837 1027 2 PRIV_MSK;
838 1028 2
839 1029 2 PLACE_HLDR = PRV$AB NAMES; ! point to start of privilege name table
840 1030 2 PRVS TO PRINT = FALSE; ! record status of buffer
841 1031 2 FORMAT_LINE ( FAOCTL_PRIVHD ); ! init buffer with header info and indentation
842 1032 2
843 1033 2
844 1034 2 WHILE (.PLACE_HLDR) <0,8> NEQ 0 DO ! Traverse down the table
845 1035 2 BEGIN
846 1036 2 PLACE_HLDR = .PLACE_HLDR + 1; ! Second byte is privilege mask
847 1037 2 PRIV_MSK = (.PLACE_HLDR) <0,8>;
848 1038 2 PLACE_HLDR = .PLACE_HLDR + 1; ! Third byte is ASCII string count
849 1039 2 SYMBOL_LEN = (.PLACE_HLDR) <0,8>;
850 1040 2
851 1041 2 IF (.PRIV_ADR) <.PRIV_MSK,1> ! Check if bit is set in quadword
852 1042 2 THEN
853 1043 2 BEGIN
854 1044 2
855 1045 2 The bit is set, put ASCII in buffer
856 1046 2
857 1047 2 PRVS TO PRINT = TRUE; ! Remember that something is in buffer
858 1048 2 FORMAT_LINE ( FAOCTL_PRIV, .PLACE_HLDR);
859 1049 2 IF IN$C_FAOBUFLN -.IN$C_FAOBUFDSC [D$C$W_LENGTH] GTR 70
860 1050 2 THEN
861 1051 2 BEGIN
862 1052 2
863 1053 2 Avoid too long a line. If it is, print what we have and
864 1054 2 start a new line with a blank header offset
865 1055 2
866 1056 2 TERMINATE_LINE ();
867 1057 2 PRVS TO PRINT = FALSE; ! Currently no privs in buffer
868 1058 2 FORMAT_LINE ( FAOCTL_PRIVHD2 );
869 1059 2 END;
870 1060 2 END;
871 1061 2
872 1062 2
873 1063 2 skip past count byte and ASCII privilege symbol
874 1064 2

```

PRINT_PRIVS

```

875 1065 3 PLACE_HLDR = .PLACE_HLDR + 1 + .SYMBOL_LEN;
876 1066
877 1067 END; ! while
878 1068
879 1069
880 1070 IF .PRVS TO PRINT ! If there is something other than the header in the buffer
881 1071 THEN TERMINATE_LINE () ! Then print it
882 1072 ELSE
883 1073 BEGIN ! otherwise reset buffer to forget about unused priv header
884 1074 INSS$FAOBUFDISC [DSCSW_LENGTH] = INSS$FAOBUFLEN;
885 1075 INSS$FAOBUFDISC [DSCSA_POINTER] = .INSS$FAOOUTBUF;
886 1076 END;
887 1077
888 1078 RETURN TRUE;
889 1079 END; ! routine print_privs

```

00FC 00000 PRINT_PRIVS:

					WORD	Save R2,R3,R4,R5,R6,R7	1011
					MOVAB	FORMAT LINE, R7	
					MOVAB	INSS\$FAOBUFDISC, R6	
					MOVAB	PRVSAB NAMES, PLACE_HLDR	1029
					CLRL	PRVS TO PRINT	1030
					PUSHAB	FAOCTL PRIVHD	1031
					CALLS	#1, FORMAT LINE	
					TSTB	(PLACE_HLDR)	1034
					BEQL	3\$	
					INCL	PLACE_HLDR	1036
					MOVZBL	(PLACE_HLDR)+, PRIV_MSK	1037
					MOVZBL	(PLACE_HLDR), SYMBOL_LEN	1039
2A	04				BBC	PRIV_MSK, @PRIV_ADR, 2\$	1041
					MOVL	#1, PRVS TO PRINT	1047
					PUSHL	PLACE_HLDR	1048
					PUSHAB	FAOCTL PRIV	
					CALLS	#2, FORMAT LINE	
					MOVZWL	INSS\$FAOBUFDISC, R0	1049
					MOVAB	70(R0), R0	
					CMPL	R0, #255	
					BGEQ	2\$	
					CALLS	#0, TERMINATE_LINE	1056
					CLRL	PRVS TO PRINT	1057
					PUSHAB	FAOCTL PRIVHD2	1058
					CALLS	#1, FORMAT LINE	
					MOVAB	1(SYMBOL_LEN)[PLACE_HLDR], PLACE_HLDR	1065
					BRB	1\$	1034
					BLBC	PRVS TO PRINT, 4\$	1070
					CALLS	#0, TERMINATE_LINE	1071
					BRB	5\$	
					MOVZBW	#255, INSS\$FAOBUFDISC	1074
					MOVL	INSS\$FAOOUTBUF, INSS\$FAOBUFDISC+4	1075
					MOVL	#1, R0	1078
					RET		1079

; Routine Size: 117 bytes. Routine Base: \$CODE\$ + 04F0

INSLIST
V04-000

PRINT_PRIVS

6 10
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 BLISS-32 V4.0-742
[INSTAL.SRC]INSLIST.B32;1

Page 33
(10)

: 890

1080 1

ouput to temporary buffer routines

```

892 1081 1 %SBTTL 'ouput to temporary buffer routines';
893 1082 1
894 1083 1 ROUTINE FORMAT_LINE (FAO_STRING, PARAMETER_LIST) =
895 1084 2 BEGIN
896 1085 2 +++
897 1086 2
898 1087 2 FUNCTIONAL DESCRIPTION:
899 1088 2 Format an ASCII string and stuff it into the output buffer.
900 1089 2 Update the buffer pointers to reflect the new stuff in the
901 1090 2 buffer.
902 1091 2
903 1092 2 INPUT:
904 1093 2 fao_string = Formatted Ascii Output control string for FAO
905 1094 2 parameter_list= List of stuff to have formatted into buffer.
906 1095 2
907 1096 2 IMPLICIT INPUT:
908 1097 2 Output buffer has been allocated and ins$faobufdesc is the
909 1098 2 descriptor for it.
910 1099 2
911 1100 2 OUTPUT:
912 1101 2 none
913 1102 2
914 1103 2 ROUTINE VALUE
915 1104 2 Success, or error status from SYS$FAOL
916 1105 2 ---
917 1106 2 LOCAL
918 1107 2 OUTLEN : WORD;
919 1108 2
920 1109 2 EXECUTE ( SYS$FAOL (.FAO_STRING, OUTLEN, INSS$FAOBUFDESC, PARAMETER_LIST));
921 1110 2 INSS$FAOBUFDESC [DSC$W_LENGTH] = .INSS$FAOBUFDESC [DSC$W_LENGTH] - .OUTLEN;
922 1111 2 INSS$FAOBUFDESC [DSC$A_POINTER] = .INSS$FAOBUFDESC [DSC$A_POINTER] + .OUTLEN;
923 1112 2 RETURN TRUE;
924 1113 1 END; ! routine FORMAT_LINE

```

! Format the buffer
! decrement space left in bu
! Point to unused space left

0000 00000 FORMAT_LINE:						
	SE		04 C2 00002	.WORD	Save nothing	: 1083
		08	AC 9F 00005	SUBL2	#4, SP	
		0000'	CF 9F 00008	PUSHAB	PARAMETER_LIST	: 1109
		08	AE 9F 0000C	PUSHAB	INSS\$FAOBUFDESC	
		04	AC DD 0000F	PUSHAB	OUTLEN	
00000000G	00		04 FB 00012	PUSHL	FAO_STRING	
	10		50 E9 00019	CALLS	#4, SYS\$FAOL	
0000'	CF		6E A2 0001C	BLBC	STATUS, 1\$	
	50		6E 3C 00021	SUBW2	OUTLEN, INSS\$FAOBUFDESC	: 1110
0000'	CF		50 C0 00024	MOVZWL	OUTLEN, R0	: 1111
	50		01 D0 00029	ADDL2	R0, INSS\$FAOBUFDESC+4	
			04 0002C 1\$:	MOVL	#1, R0	: 1112
				RET		: 1113

: Routine Size: 45 bytes, Routine Base: \$CODE\$ + 0565

: 925 1114 1

output to temporary buffer routines

```

927 1115 1 ROUTINE TERMINATE_LINE : NOVALUE =
928 1116 2 BEGIN
929 1117 3 +++
930 1118 4
931 1119 5 FUNCTIONAL DESCRIPTION:
932 1120 6     Print the contents of the output buffer to sys$output and re-initialize
933 1121 7     the descriptor of the buffer, and zero the buffer.
934 1122 8
935 1123 9 INPUT:
936 1124 10     none
937 1125 11
938 1126 12 IMPLICIT INPUT:
939 1127 13     Output buffer has been allocated and ins$faobufdesc is the
940 1128 14     descriptor for it.
941 1129 15
942 1130 16 OUTPUT:
943 1131 17     Output the contents of ins$faooutbuf to sys$output
944 1132 18
945 1133 19 ROUTINE VALUE
946 1134 20     status from $PUT
947 1135 21 ---
948 1136 22
949 1137 23 LOCAL
950 1138 24     LINE_LEN;
951 1139 25
952 1140 26 LINE_LEN = INSSC FAOBUFLN - .INSSFAOBUFDSC [DSC$W_LENGTH];
953 1141 27 TMPBUF_PTR [0,0,8,0] = .LINE_LEN;
954 1142 28 TMPBUF_PTR = .TMPBUF_PTR + 1;
955 1143 29 CH$MOVE (.LINE_LEN, .INSSFAOOUTBUF, .TMPBUF_PTR);
956 1144 30 TMPBUF_PTR = .TMPBUF_PTR + .LINE_LEN;
957 1145 31
958 1146 32
959 1147 33 INSSFAOBUFDSC [DSC$W_LENGTH] = INSSC FAOBUFLN;
960 1148 34 INSSFAOBUFDSC [DSC$A_POINTER] = .INSSFAOOUTBUF;
961 1149 35 CH$FILL ('X', INSSC FAOBUFLN, .INSSFAOOUTBUF);
962 1150 36 RETURN;
963 1151 37 END;          ! Routine TERMINATE_LINE

```

03FC 0000 TERMINATE LINE:

		59	0000'	CF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9	1115
		58	0000'	CF	9E	00007	MOVAB	INSSFAOBUFDSC, R9	
		56		69	3C	0000C	MOVZWL	INSSFAOBUFDSC, LINE_LEN	1140
	56	000000FF		56	C3	0000F	SUBL3	LINE_LEN, #255, LINE_LEN	
		00		56	90	00017	MOVB	LINE_LEN, @TMPBUF_PTR	1141
				68	D6	0001B	INCL	TMPBUF_PTR	1142
			FC	A9	D0	0001D	MOVL	INSSFAOOUTBUF, R7	1143
	00	88		56	28	00021	MOVC3	LINE_LEN, (R7), @TMPBUF_PTR	
				56	C0	00026	ADDL2	LINE_LEN, TMPBUF_PTR	1144
			FF	8F	9B	00029	MOVZBW	#255, INSSFAOBUFDSC	1147
				57	D0	0002D	MOVL	R7, INSSFAOBUFDSC+4	1148
00FF	8F			00	2C	00031	MOVC5	#0, (SP), #32, #255, (R7)	1149
		20		6E		00038			

INSLIST
V04-000

ouput to temporary buffer routines

J 10
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 Bliss-32 V4.0-742
[INSTAL.SRC]INSLIST.B32;1

Page 36
(12)

04 00039

RET

: 1151

: Routine Size: 58 bytes, Routine Base: \$CODES + 0592

: 964 1152 1

ouput to temporary buffer routines

```

: 966      1153 1 ROUTINE FORMAT_TERMINATE_LINE (FAO_STRING,PARAMETER_LIST) : NOVALUE =
: 967      1154 2 BEGIN
: 968      1155 3 +++
: 969      1156 3 FUNCTIONAL DESCRIPTION:
: 970      1157 3
: 971      1158 3     Call FORMAT_LINE to format the line, then call TERMINATE_LINE to
: 972      1159 3     terminate the line.
: 973      1160 3
: 974      1161 3 ---
: 975      1162 3 BUILTIN
: 976      1163 3     CALLG,
: 977      1164 3     AP;
: 978      1165 3
: 979      1166 3 CALLG(.AP,FORMAT_LINE);
: 980      1167 3 TERMINATE_LINE();
: 981      1168 2 RETURN;
: 982      1169 1 END;

```

0000 00000 FORMAT_TERMINATE_LINE:

93	AF	6C	FA	00002	.WORD	Save nothing	
BC	AF	00	FB	00006	CALLG	(AP), FORMAT_LINE	
			04	0000A	CALLS	#0, TERMINATE_LINE	
					RET		

: 1153
: 1166
: 1167
: 1169

; Routine Size: 11 bytes, Routine Base: \$CODE\$ + 05CC

output to temporary buffer routines

```

: 984 1170 1 ROUTINE PRINTOUT =
: 985 1171 2 BEGIN
: 986 1172 3 +++
: 987 1173 4
: 988 1174 5 FUNCTIONAL DESCRIPTION:
: 989 1175 6 Print the contents of the temporary buffer to sys$output
: 990 1176 7
: 991 1177 8 INPUT:
: 992 1178 9 none
: 993 1179 10
: 994 1180 11 IMPLICIT INPUT:
: 995 1181 12 Output buffer has been allocated and ins$faobufdesc is the
: 996 1182 13 descriptor for it.
: 997 1183 14
: 998 1184 15 OUTPUT:
: 999 1185 16 Output the contents of ins$faobuf to sys$output
1000 1186 17
1001 1187 18 ROUTINE VALUE
1002 1188 19 status from $PUT
1003 1189 20 ---
1004 1190 21
1005 1191 22 LOCAL
1006 1192 23 TMPBUF_USELEN,
1007 1193 24 STATUS;
1008 1194 25
1009 1195 26 TMPBUF_USELEN = .TMPBUF_PTR - .TMPBUF;
1010 1196 27 TMPBUF_PTR = .TMPBUF;
1011 1197 28
1012 1198 29 WHILE .TMPBUF_PTR - .TMPBUF LSS .TMPBUF_USELEN DO
1013 1199 30 BEGIN
1014 1200 31 LOCAL
1015 1201 32 SIZE;
1016 1202 33
1017 1203 34 SIZE = .(.TMPBUF_PTR) <0,8,0>;
1018 1204 35 INSG_OUTTAB [RAB$W_RSZ] = .SIZE;
1019 1205 36 INSG_OUTTAB [RAB$L_RBF] = .TMPBUF_PTR+1;
1020 1206 37 EXECUTE ($PUT (RAB = INSG_OUTTAB));
1021 1207 38 TMPBUF_PTR = .TMPBUF_PTR + 1 + .SIZE;
1022 1208 39 END;
1023 1209 40
1024 1210 41 RETURN TRUE;
: 1025 1211 42 END; ! Routine PRINTOUT

```

.EXTRN SYSSPUT

003C 0000 PRINTOUT:

	55	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5	: 1170
	54	0000	CF	9E	00009	MOVAB	INSG_OUTTAB+34, R5	:
53	64	FC	A4	C3	0000E	MOVAB	TMPBUF_PTR, R4	:
	64	FC	A4	D0	00013	SUBL3	TMPBUF, TMPBUF_PTR, TMPBUF_USELEN	: 1195
	51		64	D0	00017	MOVL	TMPBUF, TMPBUF_PTR	: 1196
50	51	FC	A4	C3	0001A	MOVL	TMPBUF_PTR, R1	: 1198
	53		50	D1	0001F	SUBL3	TMPBUF, R1, R0	:
			22	18	00022	CMPL	R0, TMPBUF_USELEN	:
						BGEQ	2\$:

INSLIST
V04-000

ouput to temporary buffer routines

M 10
16-Sep-1984 01:54:25
14-Sep-1984 12:35:38

VAX-11 Bliss-32 V4.0-742
[INSTAL.SRC]INSLIST.B32;1

Page 39
(14)

	52		61	9A	00024	MOVZBL	(R1), SIZE	: 1203
	65		52	80	00027	MOVW	SIZE, INSSG_OUTTAB+34	: 1204
06	A5	01	A1	9E	0002A	MOVAB	1(R1), INSSG_OUTTAB+40	: 1205
		DE	A5	9F	0002F	PUSHAB	INSSG_OUTTAB	: 1206
00000000G	00		01	FB	00032	CALLS	#1, SYSSPUT	
	0D		50	E9	00039	BLBC	STATUS, 3\$	
50	64		52	C1	0003C	ADDL3	SIZE, TMPBUF_PTR, R0	: 1207
	64	01	A0	9E	00040	MOVAB	1(R0), TMPBUF_PTR	: 1198
			D1	11	00044	BRB	1\$: 1210
	50		01	D0	00046	MOVL	#1, R0	: 1211
			04	00049	3\$:	RET		

; Routine Size: 74 bytes, Routine Base: \$CODE\$ + 05D7

: 1026	1212	1	
: 1027	1213	1	
: 1028	1214	1	END
: 1029	1215	0	ELUDOM

! Module inslist

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	12	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$OWNS	144	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	672	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODE\$	1569	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
. ABS .	0	NOVEC, NOWRT, NORD, NOEXE, NOSHR, LCL, ABS, CON, NOPIC, ALIGN(0)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	74	0	1000	00:01.8

COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:INSLIST/OBJ=OBJ\$:INSLIST MSRC\$:INSLIST/UPDATE=(ENH\$:INSLIST)

; Size: 1569 code + 828 data bytes
; Run Time: 00:31.3

INSLIST
V04-000

ouput to temporary buffer routines

N 10
16-Sep-1984 01:54:25

VAX-11 Bliss-32 V4.0-742

Page 40

; Elapsed Time: 01:38.9
; Lines/CPU Min: 2326
; Lexemes/CPU-Min: 19941
; Memory Used: 261 pages
; Compilation Complete

0189

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY